Reverse Engineering Questionnaire[†]

Kostas A. Kontogiannis
School of Computer Science
McGill University
kostas@cs.mcgill.ca

Scott R. Tilley
Department of Computer Science
University of Victoria
stilley@csr.uvic.ca

Abstract

This is a questionnaire on program understanding and reverse engineering. It may be filled out manually or on-line. The results of the questionnaire will be used to guide the research of the two authors, both of whom are Ph.D. students working in this area. Copies of the resulting report will be mailed to all who participate, and a summary of the results will be published in an appropriate forum.

Keywords: program understanding, reverse engineering, software maintenance

Purpose of this questionnaire

There are numerous reverse engineering tools and techniques available in the marketplace, including both commercial offerings and research prototypes. The purpose of this questionnaire is to gather information on the use of such tools by software professionals in academia, government, and industry. The results of the questionnaire will be used to guide the research of the two authors, both of whom are Ph.D. students working in the area of program understanding and reverse engineering.

[†]This work was supported in part by the British Columbia Advanced Systems Institute, IBM Canada Ltd., the IRIS Federal Centres of Excellence, McGill University, the Natural Sciences and Engineering Research Council of Canada, the Science Council of British Columbia, and the University of Victoria.

Submitted to *ACM SIGSOFT Software Engineering Notes*, January 1994.

We are attempting to gather realistic data on the use of program understanding and reverse engineering tools in the software community. We are also soliciting information on the wishes of users for the tools of tomorrow, and are interested in their thoughts on the integration of such technologies into their current software environments and processes.

If you have experience in software development and/or maintenance (and this includes as a project manager, technical writer, tester, and so on), we would very much appreciate your responses to the questions on this survey. If you do not feel qualified to answer this questionnaire, please pass it along to someone who can answer the questions. Also, if you know someone who we might ask to participate, please send us his or her name and address. We will mail a copy of the resulting report to all who participate, and it will be published in an appropriate forum.

While some of the questions ask for information about your personal history, we will only be addressing group data; any information you provide will be kept confidential. If answers to some of the questions are confidential or unknown, just make a note of that and please try to answer the other questions. If you have any questions or comments about the questionnaire, please direct them to the second author.

The questionnaire consists of five parts: General information, tools, methods, integration issues, and miscellaneous. It takes approximately forty minutes to complete all forty questions.

Reverse engineering

Software evolution refers to the on-going enhancements of existing software systems, involving both development and maintenance. As software ages and *evolves*, the task of maintaining it becomes more complex and more expensive. Several areas have been identified as critical to supporting software evolution; *recapture* technologies are one of them. Recapture technologies attempt to recover the original design in existing software systems.

One of the most promising approaches to the problem of software evolution is *program understanding.* It has been estimated that fifty to ninety percent of evolution work is devoted to program understanding. For legacy software systems,[1] this is a very difficult task. One way of augmenting the understanding process is through *reverse engineering.*

Reverse engineering is the process of extracting system abstractions and design information from existing software systems. This information can then be used for subsequent development, maintenance, project management, re-engineering, or reuse purposes. The process involves the identification of software artifacts in a particular subject system, and the aggregation of these artifacts to form more abstract system representations.

During reverse engineering the subject system is not altered, although additional information about it is generated. In contrast, the process of re-engineering typically consists of a reverse engineering phase, followed by a forward engineering or reimplementation phase which alters the subject system.

[1] Legacy software systems are those that are 10-25 years old and often in poor condition because of prolonged maintenance.

Filling out the questionnaire

You may complete this questionnaire either manually or on-line.

Manually

If you wish, you may simply fill in the questionnaire attached to this paper. After filling it in, kindly mail the completed form to the second author; the full address is given at the end of the questionnaire.

On-line

This questionnaire is being distributed electronically to various newsgroups on the Internet. It is also available via anonymous ftp from tara.uvic.ca (128.189.88.123) in the directory pub/re-q as the shar file re-q.shar.

To unbundle the questionnaire, type:

```
sh -f re-q.shar -c
```

This will create the directory ./re-q, which contains the questionnaire forms, executable csh script, and the file you are currently reading (re-q.ps). Please note that the executable script is **not** a "trojan horse," nor will it harm your system in any way. It simply invokes your favorite editor on the five parts of the questionnaire, bundles the results together, and e-mails the resultant file to the second author. All work is done in /tmp.

To fill out the questionnaire on-line, just type `re-q` in the ./re-q directory and follow the instructions. The results will be e-mailed to the second author.

Thank you in advance for your participation!

Questionnaire

For answers such as Importance or Usage, which require a numerical rating, please use the following sliding scale: 1=low ⇒ 5=high. Please fill in empty table entries with your own information. If you run out of room while answering any of the questions, please attach additional sheets of paper to your completed questionnaire.

Part 1: General Information

Question 1.1: Background information:

Name	:
Work Address	:
Tel.	:
Fax	:
E-mail	:
Age (optional)	:

Question 1.2: What is your profession?

Question 1.3: What is your current work assignment?

Question 1.4: How long have you been involved with software development and/or maintenance? (This may include project management, testing, documentation, and so on.)

Question 1.5: What is the approximate size of programs you maintain (in KLOC)?

Question 1.6: What programming languages are these programs written in?

Question 1.7: Approximately how many end-user applications have you developed?

Question 1.8: What is your support environment?

- Operating system:
- Hardware platforms:
- Supporting or specialized software:
-
-

Question 1.9: Are there aspects of your software maintenance tasks which are unique, such as special-purpose and/or proprietary programming languages, thus making the use of commercial off-the-shelf tools difficult?

Question 1.10: Is there anything else you would like to tell us about yourself?

Part 2: Tools

Question 2.1: Please list your personal "top five" most important tools you use for software maintenance, reverse engineering, and/or program understanding.

	Tool	Vendor
1		
2		
3		
4		
5		

Question 2.2: What do you like the most about these tools?

Question 2.3: What do you like the least about these tools?

Question 2.4: For the tools you like and use, how well do they address your software maintenance needs? How long have you been using them? Could you do your job without them?

Question 2.5: What are important tool features for your work? (Customizability may imply a scripting language; flexibility may mean the tool is applicable to multiple domains; scalability may mean the tool is applicable to programs of $O(10^6)$ LOC.)

Feature	Importance
Customizability	
Flexibility	
Scalability	

Question 2.6: Which of the following visual representations of source code are the most useful to you?

Visual Representation	Importance
Abstract syntax trees	
Control flow diagrams	
Cross-reference listing	
Data flow diagrams	
Data structure diagrams	
Decision tables	
Design notations	
Multiple views	
Petrinets	
Process comm. (CCS/CSP)	
Pretty printers	
Pseudocode	
Slicing and segmentation	
State transition diagrams	
Structure charts	

Question 2.7: Which techniques and tools do you use (or would like to use) for locating code fragments during maintenance?

Method / Tool	Usage
Heuristics based on identifier names	
Uses of variables	
Regular expressions	
Data / Control dependencies	
Preconditions / Postconditions	
Domain mapping	
Informal information	
Naming conventions	

Question 2.8: Do you rely on dynamic analysis?

Method / Tool	Usage
Debuggers	
Profilers	
Test-Case coverage analyzers	

Question 2.9: Do you rely on static analysis?

Method / Tool	Usage
Def / Use chains	
Memory allocation / Deallocation	
Value range analysis	

Question 2.10: Do you use software metrics?

Metric	Usage
Cyclomatic Complexity	
Coupling/Cohesion	
Function Point	
Information Flow	
Hierarchy depth (O-O systems)	
Class lattice structure	

Question 2.11: Which change and impact analysis method or tool do you use (or would like to use)?

Method / Tool	Usage
Cross reference listings	
Slicing	
Data / Control dependencies	

Question 2.12: Which tools do you use for documentation or automatic (re)documentation? If none, which tools would you like to use?

Method / Tool	Usage
Word processor	
Drawing tools	
Text editor	

Question 2.13: What do you rely on for tool news? (please check with an X)

Conferences	Use
COMDEX	
CSM	
ICSE	
International Conference on CASE	
OOPSLA	
Program Comprehension Workshop	
Reverse Engineering Forum	
SIGSOFT	
Working Conf. on Reverse Engineering	
Forums	
Compuserve	
Internet news	
Internal bulletin boards	
Journals	
CACM	
IEEE Computer	
IEEE Software	
Software Maint.: Research & Practice	
TSE	
TOSEM	
Magazines	
Byte	
Datamation	
PC Magazine	
Software Development	
UNIX World	
Organizations	
ACM	
IEEE	
USENIX	

Part 3: Methods

Question 3.1: Please describe a typical reverse engineering scenario for you.

Question 3.2: Do you use formal methods? If so, which ones?

Question 3.3: How might reverse engineering and/or program understanding tools help you in your software maintenance tasks?

Question 3.4: How do you reverse engineer event-driven (distributed, triggered, or call-back) programs? (If applicable to you.)

Question 3.5: Where do you think automation is possible in reverse engineering?

Question 3.6: If you are not currently using reverse engineering and/or program understanding tools to aid your software maintenance work, why not? (Please check with an X all applicable reasons.)

Reason	True?
No (perceivable) benefits	
Poor method support	
Cost	
No management backing	
Current methods sufficient	
Don't know market well enough	
Too much training needed	
Tools fail on "real world" code	
Lack of customizability	
Difficult to integrate	
Hard to quantify gains	
Poor user interface	
Poor documentation	
Multiuser support	
Too much effort to start	

Question 3.7: What is the most effective way of focusing attention and discovering information for you?

Strategy	Effectiveness
Top-down	
Bottom-up	
Mixed	

Part 4: Integration Issues

Question 4.1: Please describe your current maintenance process.

Question 4.2: What impediments do you see to integrating new reverse engineering and program understanding technology into your current software processes?

Question 4.3: What changes (managerial and otherwise), would be required to successfully integrate this technology into your current processes and environment?

Question 4.4: What do you see as the single largest stumbling block to you (or your company) embracing reverse engineering technology?

Part 5: Miscellaneous

Question 5.1: What is your opinion on the current state-of-the-art/state-of-the-practice in reverse engineering and program understanding tools and techniques?

Question 5.2: Do you think today's tools adequately address your needs? (Please elaborate.)

Question 5.3: What do you think is important for a tool to provide?

Question 5.4: What do you think the future holds for this area? Will reverse engineering exist twenty years from now?

Question 5.5: Any final comments about program understanding and reverse engineering?

Question 5.6: What might be done to improve this questionnaire?

Thank you for taking the time to complete this questionnaire. Please mail the completed form to:

Scott R. Tilley
Department of Computer Science
University of Victoria
P.O. Box 3055
Victoria, BC
Canada V8W 3P6

Thanks! [KAK, SRT]